A ROS-integrated API for the KUKA LBR iiwa collaborative robot ⋆

Saeid Mokaram * **Uriel Martinez-Hernandez** ** **Jonathan M. Aitken** * **Iveta Eimontaite** * **David Cameron** * **James Law** * **Joe Rolph** * **Ian Gwilt** * **Owen McAree** *

* *Sheffield Robotics, University of Sheffield, UK (e-mail: s.mokaram, jonathan.aitken, i.eimontaite, d.cameron, o.mcaree@sheffield.ac.uk, j.law@sheffield.ac.uk,j.rolph, i.gwilt@shu.ac.uk).*
** *Institute of Design, Robotics and Optimisation, School of Mechanical Engineering, Faculty of Engineering, University of Leeds (e-mail: U.Martinez@leeds.ac.uk).*

Abstract: As Industry 4.0 develops the manufacturing sector is increasingly looking towards new methodologies to employ robots alongside line workers. This paper presents an Application Programming Interface (API) for the KUKA Intelligent Industrial Work Assistant (iiwa) Lightweight Robot (LBR). This API builds upon the safety embedded within the KUKA iiwa to allow close working and interaction with operators. It brings the functionality into the Robot Operating System (ROS), which provides a distributed development environment allowing multiple new modalities of devices to interface easily. This paper presents an example application developed within the API which has allowed a large-scale interactive participant experiment of wholly inexperienced users to be conducted using the KUKA iiwa.

Keywords: Computer Programming, Robotics, Industrial Robots, Agile Manufacturing, Robot Programming

1. INTRODUCTION

The manufacturing sector is poised to undergo considerable change over the next decade. Driven by initiatives such as Industry 4.0, the Digital Agenda, and the Internet of Things, the introduction of new technologies and further digitalisation will lead to highly connected, and integrated workplaces. These changes will lead to new ways of working, and open up opportunities for process flexibility. In particular, developments in robotics will enable humans and robots to work collaboratively, maximising the benefits of manual and automated processes (Pawar et al., 2016).

This shift towards human-robot co-working is enabled by the recent development of collaborative robots, including the KUKA LBR iiwa. Such *cobots* are designed to operate alongside human users in shared environments without safety caging; backdrivable motors and compliant controllers allowing humans to physically interact with the robots without harm. Whilst early adoption focuses on robots working un-caged in human-occupied spaces as assistive tools with little interaction, the full potential of this technology will only be realised through symbiotic human-robot collaborative processes.

A major aim for future manufacturing is greater flexibility to support smaller batch sizes and more customisation. Whereas existing automated processes are highly repetitive, and difficult to reconfigure for new products or tasks, collaborative robots will support much greater variability through task switching. The added complexity of flexible processes and co-working with robots will require the up-skilling of the human workforce, and highly intuitive interfaces to support more variability in worker roles.

To achieve these aims, greater integration of workers, robots, and systems are required. This requires development of robot control interfaces that can safely and flexibly connect to sensors and gather information from external sources; that can be reprogrammed by non-experts; and that can provide intuitive information to users. In this paper, we describe an interface for the KUKA LBR iiwa, available online [1], which we have developed to support our experimental research work, and is now supporting development of new industrial processes. The interface enables control and communication via the Robot Operating System (ROS), but with minimal installation requirement, and without compromising the inbuilt safety features.

1.1 The Growing Importance of Co-Working with Robots

Collaboration between humans and robots has been observed in the manufacturing sector since 1940s. Applica-

⋆ This work has been supported by the EPSRC Centre for Innovative Manufacturing in Intelligent Automation grant "Assessing Graphical Robot Aids for Interactive Co-working (A-GRAfIC)", and the University of Sheffield Impact, Innovation and Knowledge Exchange grant "Human Robot Interaction Development". Equipment has been provided under the EPSRC Capital for Great Technologies: Robotics and Autonomous Systems. All authors are members or alumni of Sheffield Robotics, which facilitated this work.

[1] `https://github.com/jonaitken/KUKA-IIWA-API`

tions such as sorting, cutting and painting were initially based on simple control and communication methods; on-off switches, analog joysticks and unidirectional communication. Over time, with advances in technology and artificial intelligence methods, co-working with robots has gradually become more robust and safe, offering natural control and communication methods (Sheridan, 1997).

Multiple key requirements have been identified for co-working with robots. Dialog is a factor needed for effective communication and exchange of information between humans and robots. This communication process, based on multiple modalities, e.g., visual and touch sensing, can be used to ask questions and evaluate the quality of tasks (Fong et al., 2003). To achieve these intelligent co-working platforms, knowing 'what type of information?', 'what medium of communication?' and 'when communication should occur?' is essential to enrich the quality of collaboration between humans and robots (Kaupp et al., 2010).

Safety in human-robot interaction is also a crucial factor, that needs of robust frameworks for collaborative communication, control and monitoring (Thomas et al., 2011). A message-based architecture composed of distributed modules was proposed for teleoperation, which included safety and sensor management modules, wide range of interfaces for communication and supported various degrees of cooperation and autonomy (Fong et al., 2001). A human-robot communication framework, based on integration of data from multiple sensory sources, allowed to exchange information and achieve a task collaboratively (Kaupp et al., 2010). This framework was developed using probabilistic robotics representation to infer when to transmit specific communications between humans and robots. Coordination and control of human and robot actions through dialog was addressed by the design of an operating system for human-robot interaction (Fong et al., 2006). This system used an agent-based paradigm, that supported a variety of user interfaces, task-oriented dialog, resource and interaction management, and integration of robots through an Application Programming Interface (API).

Over time, as robots stop being passive tools for humans to use and become more sophisticated and automated co-working partners, the relationship between humans and robots will change to start resembling the interaction between two individuals (Ososky et al., 2013). In addition, the current shift in industry for manufacturing tasks to incorporate human-robot co-working increases the need for improved interfaces to make this interaction more efficient. The level of autonomy, complexity and safety measures will continue to increase, yet to enable true collaboration robots will also need to gain the confidence of human operators Cameron et al. (2015). These issues will be exacerbated by the introduction, and up-skilling, of workers without robotics experience.

2. KUKA IIWA LBR

The KUKA iiwa is a lightweight industrial robotic arm with seven axes. Each of its joints is equipped with torque sensors as well as a position sensor. Sensory data enables the use of impedance control in addition to position control, thus making it possible to implement compliant behaviors. Highly accurate measurements, with down to millisecond update intervals, enables the robot to react very quickly to process forces and makes it particularly suitable for interaction with humans. The KUKA iiwa can be programmed for a variety of tasks trough "KUKA Sunrise control technology". This comprises "KUKA Sunrise OS" control software which can execute programs in JAVA as the programming language on "KUKA Sunrise Cabinet" control hardware. Although Java is a flexible and common language, an in depth knowledge about the Sunrise system is required for programming the robot and utilising its functionality.

Out-of-the-box, the robot can only be controlled through a Java-based program and all the sensory data or information related to the ongoing task is only available locally on the robot control system or the KUKA Smartpad. Even though control of the robot and access to the task information by other systems in a network can be made possible by opening a network socket in the controlling program, the desire for compatibility with ancillary systems and languages has lead us to develop our own interface.

2.1 ROS

ROS (Quigley et al., 2009) provides a unified platform independent of languages and platforms for publishing and subscribing to data streams. It also comprises a large collection of commonly used functionality and applications for robot software development such as hardware drivers, robot models, simulation tools, data-types, planning, perception and other algorithms. It provides a useful architecture for developing and deploying robot systems, which can be easily modeled using graph-based techniques (Aitken et al., 2014).

2.2 Alternative APIs

There are several existing alternative APIs available for the KUKA LBR iiwa. Each has been built with a slightly different focus and consequently is customised to its own domain. In this section we provide a brief summary of the most prominent.

ROS Industrial is a working group developing interfaces aimed at widescale industrial usage, typically the framework focuses on the larger capacity arms that are part of the KUKA range (Edwards and Lewis, 2012). At present this is only an experimental package so it is subject to regular alterations, however, the main focus of the API is not on co-working, but capturing more general industrial use.

Khansari-Zadeh and Khatib (2015) and Virga et al. (2016) focus on the interaction between operators and the robot. Khansari-Zadeh and Khatib (2015) focuses on learning actions from human demonstration, displaying different impedance to motions for critical parts of the exercise. Virga et al. (2016) investigates force-compliant motion within the medical domain. Both of the APIs produced require specific components installing upon the KUKA iiwa, which require modification of the operational parameters on board the KUKA Sunrise controller, which either change the modes of operation or require custom installation of third-party libraries.

The API developed within this paper is a simple, stand-alone application, which can be placed on the KUKA Sunrise controller, and provides functionality without necessitating any modification of the control unit. It allows direct integration with ROS, without requiring any configuration. This enables full compatibility with the Robot Systems Toolbox in Matlab and Simulink [2], which widens the choice of development platforms for API users and allows the inclusion of model-based design as a choice for verifying potential applications (McAree et al., 2016).

3. BUILDING THE API

The API developed within this paper is designed to be simple and interface to ROS to provide an easy platform for development.

3.1 API Architecture

The API architecture focuses on the breaking out the functionality that would normally be available within KUKA Sunrise controller run on the Smartpad.

The architecture can be viewed to extend the capability of the KUKA LBR iiwa, using the generic structure shown in Figure 1. The API exposes an interface to operation on a network of machines. This allows different sensing methods and extra computing resources to be easily deployed and exploited in operations of the KUKA LBR iiwa.

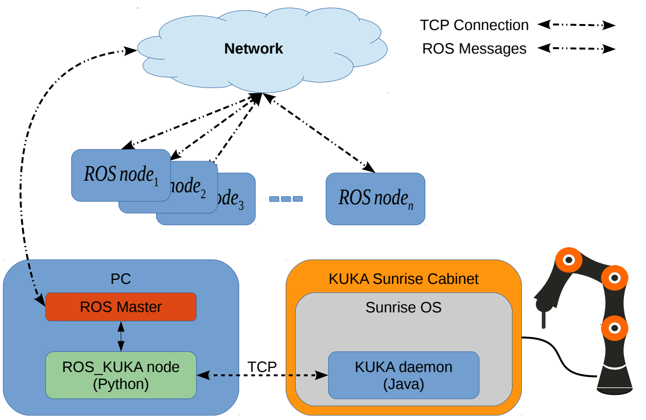

Fig. 1. *Components of the KUKA ROS interface architecture.*

The KUKA server also handles some low-level, but generic and critical controlling tasks, such as collision detection. Handling such safety-critical motion controls locally on the Sunrise system is necessary as it provides maximum reliability by using code developed to strict safety standards. This ensures the underlying operation of the robot is still safe, and the API presented in this paper forms a wrapper which utilizes these underlying safety-critical protocols.

A ROS-KUKA node is also implemented in the Python scripting language, which is a ROS node and plays an intermediate interface between the KUKA server and ROS master. It *subscribes* to the commands coming from other controlling ROS nodes and passes them to the KUKA server. Similarly it receives sensory and status information from the KUKA server and *publishes* them under their specific *Topics*. Running this ROS-KUKA node externally on a separate computer, rather than on the KUKA Sunrise OS, means that no modification is required of the KUKA Sunrise Cabinet. Therefore the safety protocol provided by KUKA is preserved, and functionality is added rather than altered. Instead of installing a third party software such as ROS on the Sunrise system, the KUKA iiwa ROS interface can be set up with just a standard Sunrise-based application. The topics available for use within ROS available through the API are shown in Table 1.

The kuka_command topic is then used to send instructions through the API to the KUKA LBR iiwa. A list of commands and descriptions is shown in Table 2.

3.2 Safety within API

It is important that the native safety of the KUKA iiwa Lightweight Robotic Arm is preserved. The arm has been developed as a robot for co-working, especially with the available compliance modes that provide the ability to work without a safety cage (Shepherd and Buchstab, 2014; Kirchner et al., 2015). This provides exceptional capability as a user is able to physically move the robot arm, whilst exposed to a level of risk deemed safe.

The API developed uses the standard safety settings on the KUKA iiwa Lightweight Robotic Arm, acting as part of a subsumption architecture Brooks (1986). Ultimately the safety settings on the KUKA Sunrise Cabinet always interpret the control provided from ROS, whilst checking any command to ensure the arm remains within the valid operating region. If this region is not specified, or an inaccessible position is demanded, the KUKA controller will not permit movement of the arm.

By relying on the inbuilt safety functionality within a subsumption architecture our KUKA iiwa API maintains safety whilst extending the capability of the system. The safety functionality within the KUKA Sunrise Controller is separated from the ROS interface; separating the reliance and ensuring the high-integrity safety elements operate as intended by the manufacturer, and without contamination. This enables a standard risk-assessment to be conducted for experimental work, which leverages the inbuilt safety and compliance supplied with the KUKA iiwa. This relies on the robot operating with reduced velocity in T1-mode, or with limited velocity when in compliance mode. Standard operating procedures dictate that the dead man's switches are depressed, and the play button activated by an experimental observer in order to activate the robot.

With these features in place a participant is able to co-work in very close proximity to the robot without needing special instructions, training or a safety cage, as the API and KUKA Sunrise Controller enables "safe-by design" application development.

3.3 Exploiting the API

The API has been used to develop the A-GRAfIC application. This experiment is designed to evaluate a graphical language developed to aid a robot co-worker. Eimontaite et al. (2016) describes the scenario where a co-worker is

[2] https://www.mathworks.com/products/robotics/

Table 1. Topics available through KUKA iiwa ROS interface. Update on a frequency of 10Hz

Topic Name and Description	Description	Example
JointPosition [A1, A2, A3, A4, A5, A6, A7] time	Joint position (in degrees), reading time-stamp	JointPosition [0.0, 0.17, 0.0, 1.92, 0.0, 0.35, 0.0] 1459253274.1
ToolPosition [X, Y, Z, A, B, C] time	Tool/end effector position (in cartesian space), reading time-stamp	ToolPosition [433.59711426170867, 0.028881929589094864, 601.4449734558293, 3.1414002368275726, 1.0471367465304213, 3.141453681799645] 1459253274.11
ToolForce [X, Y, Z] time	External force on tool/end effector in different directions, reading time-stamp	ToolForce [13.485958070668463, 0.3785658886199012, 5.964988607372689] 1459253274.11'
ToolTorque [A, B, C] time	External torque on tool/end effector in different directions, reading time-stamp	ToolTorque [13.485958070668463, 0.3785658886199012, 5.964988607372689] 1459253274.11'
JointAcceleration Float time	Joint acceleration value, reading time-stamp	JointAcceleration 0.4 1459253274.11'
JointVelocity Float time	Joint velocity value, reading time-stamp	JointVelocity 1.0 1459253274.11'
JointJerk Float time	Joint Jerk value, reading time-stamp	JointJerk 1.0 1459253274.11'
isCompliance Boolean time	Robot compliance status, reading time-stamp	isCompliance off 1459253274.11'
isReadyToMove Boolean time	Robot motion status; True if the robot can move or if the robot performed all the motion in its queue, reading time-stamp	isReadyToMove true 1459253274.11'
isCollision Boolean time	True if a collision has detected.	isCollision false 1459253274.11'
isMastered Boolean time	True if is mastered, reading time-stamp	isMastered true 1459253274.11'
isJointOutOfRange Boolean time	True if any joint is out of its range.	isJointOutOfRange false 1459253274.11'
OperationMode String time	Operation Mode T1/T2/AUT	OperationMode T1 1459253274.11'

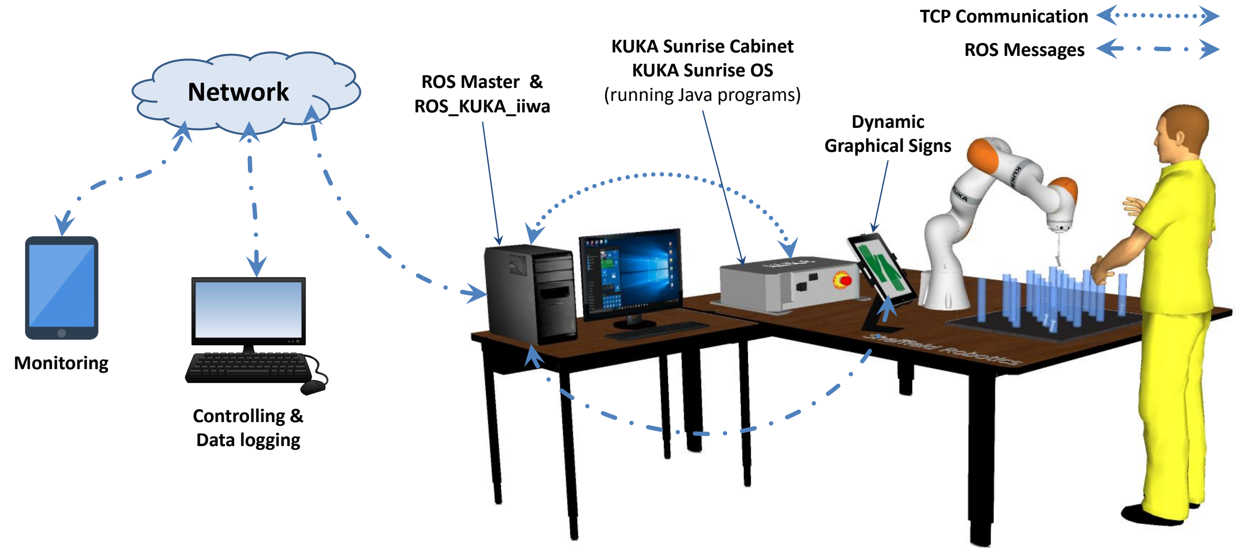

Fig. 2. *An example of the general structure of the KUKA iiwa ROS interface to Multiple Mobile Devices, exploited in the Assessing Graphical Robot Aids for Interactive Co-working (A-GRAfIC) project.*

required to remove bolts from perspex tubes, an overview is summarised in Figure 3, which is broken down further in Algorithms 1 and 2 to show how the API is used directly. The robot is aware of the tube locations, however, cannot determine which contain bolts. The co-worker decides which tube to remove a bolt from, and moves the robot in compliant mode to the location. The robot switches to autonomous modes, retrieves the bolt and presents it to the co-worker. The task is then repeated for further bolts.

The A-GRAfIC experiment was developed using the API described in Section 3. The overview of the process is described by the algorithms 1 and 2. Algorithm 1, sets the KUKA iiwa to the home position and sets up the compliance of the arm in only the x-y plane so that the operator can manually push it into the required pick up position. Algorithm 2 allows the operator a short time to change their mind, detecting if they apply a force to the arm in a 3s window, if they do compliance is turned back on to allow them to re-select a position. Once a position is confirmed, the arm takes full control and undertakes a motion to pick up the bolt and return it to the operator. The process is then repeated.

4. CONCLUSIONS

This paper has detailed the development of an API for the KUKA iiwa LBR, running KUKA Sunrise, compati-

Table 2. Topics available through KUKA iiwa ROS interface.

Topic Name and Description	Description	Example
setJointAcceleration F	Setting/changing the joint acceleration value	'setJointAcceleration 0.4'
setJointVelocity F	Setting/changing the joint velocity value	'setJointVelocity 1.0'
setJointJerk F	Setting/changing the joint jerk value	'setJointJerk 1.0'
setCartVelocity F	Setting/changing the cartesian velocity (mm/s) value	'setCartVelocity 100'
setPosition A1 A2 A3 A4 A5 A6 A7	Moving the robot arm based on joint position. Angular values (in degrees) of type float can be replaced in A1-7. In case any axis doesnt need to be moved, a - can be used instead of a value. The example assigns new positions for each axis except A2 which doesnt move.	'setPosition 0 - 0 -100 0 60 0'
setPositionXYZABC X Y Z A B C ptp/lin	Moving the robot end effector in the robot cartesian space. Point-to-point (ptp) or linear (lin) motion can be selected. This moves the robot end effector to a particular location [x,y,z] orientation [a,b,c] (values in float). In case any parameter doesnt need to be changed, a - can be used instead of a value.	'setPositionXYZABC 700 290 - -180 0 -180 lin'
MoveXYZABC X Y Z A B C	Moving the robot end effector in the cartesian space with linear (lin) motion only. This moves the robot end effector in certain direction [x,y,z] and/or orientation [a,b,c] for the given values (in mm and degrees).	MoveXYZABC 10 20 0 30 0 0
MoveCirc X1 Y1 Z1 A1 B1 C1 X2 Y2 Z2 A2 B2 C2 BlendingOri	Moving the robot end effector in a arch/circular motion from its current position passing from a first one ([x1 y1 z1 a1 b1 c1]) to a second position ([x2 y2 z2 a2 b2 c2]) with a given blending value.	MoveCirc 700 0 290 -180 0 -180 710 0 300 -180 0 -180 0.1
setCompliance X Y Z A B C	Activates the robot Compliance mode with particular stiffness in each x,y,z,a,b,c. The given example activates the Compliance with a very low stiffness in x and y cartesian plain only.	'setCompliance 10 10 5000 300 300 300'
resetCompliance	Deactivates the robot Compliance mode.	resetCompliance
setCartImpCtrl X Y Z A B C Damping	Activates the robot cartesian impedance control mode with particular impedances in each x,y,z,a,b,c. The given example activates the cartesian impedance control with a very low impedance in z cartesian axis only.	'setCartImpCtrl 5000 5000 100 300 300 300 1.0'
resetCartImpCtrl	Deactivates the robot cartesian impedance control mode.	resetCartImpCtrl
resetCollision	Resets a Collision if any collision was detected.	resetCollision
forceStop	Stops the robot and removes all the robot motion queue.	forceStop
setWorkspace xmin ymin zmin xmax ymax zmax	Defining a cubic workspace boundaries.	setWorkspace 100 -300 0 600 300 500
setTool n	Switching between a finite number (N=4) of predefined tools. Tool 1 is selected by default.	setTool 2
sleep T	Suspending execution for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time.	sleep 2.5

Algorithm 1 goToStart() pseudocode

```
setJointAcceleration ← 0.4
setJointVelocity ← 1.0
setJointJerk ← 1.0′
Set initial position as point to point movement
setPositionXYZABC ← 700 0 290 -180 0 -180 ptp
while ToolPositionError > 10 do
    Wait until tool is in required position
end while
while No force is applied to tool do
    Wait until co-worker pushes tool
end while
Set compliance in x-y plane
setCompliance ← 10 10 5000 300 300 300
```

ble with ROS, and providing control capability across a distributed network.

The A-GRAfIC demonstrator is an indication of the flexibility that the API provides. The operational modes of the KUKA iiwa can be quickly switched from fully-compliant to full autonomy programatically from within a distributed network around the robot arm. This allows full sensor suites to be quickly deployed and integrated into the system, providing capability but preserving safe operation, as the KUKA safety protocol is embedded within the API.

5. ACKNOWLEDGEMENTS

The work in this paper has been supported by the Engineering and Physical Sciences Research Council (EPSRC) through the projects "Assessing Graphical Robot Aids for Interactive Co-Working (A-GRAfIC)" (EPSRC Centre for Innovative Manufacturing in Intelligent Automation Feasibility Study) and "Developing Visual Signs and Symbols to Aid Human Robot Interaction Scenarios" (EPSRC Impact Accelerator Account).

REFERENCES

Aitken, J.M., Veres, S.M., and Judge, M. (2014). Adaptation of system configuration under the robot operating system. *IFAC Proceedings Volumes*, 47(3), 4484–4492.

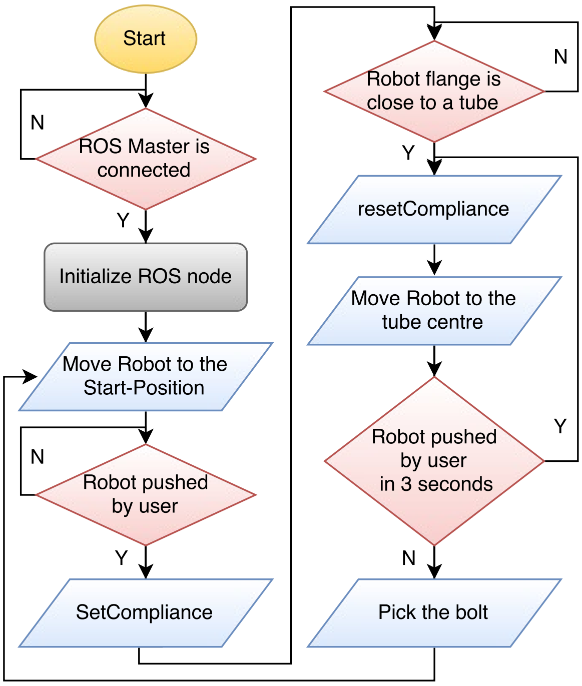

Fig. 3. Experimental Process

Algorithm 2 Pick up bolt pseudocode

Turn off compliance
resetCompliance
Center on the tube
setPositionXYZABC $\leftarrow x_p\ y_p$ *290 -180 0 -180 ptp*
while **do**$3sTimerActive$
  if Co-Worker Applies Force to the Arm **then**
    Turn on compliance
    setCompliance $\leftarrow$ *10 10 5000 300 300 300*
  end if
end while
Turn off compliance
resetCompliance
Center on the tube
setPositionXYZABC $\leftarrow x_p\ y_p$ *290 -180 0 -180 ptp*
Lower into the Tube as a linear movement
setPositionXYZABC $\leftarrow x_p\ y_p$ *120 -180 0 -180 lin*
Rise from the Tube as a linear movement
setPositionXYZABC $\leftarrow x_p\ y_p$ *290 -180 0 -180 lin*
Return to home position presenting bolt
goToStart()

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), 14–23. doi:10.1109/JRA.1986.1087032.

Cameron, D., Aitken, J.M., Collins, E.C., Boorman, L., Chua, A., Fernando, S., McAree, O., Martinez-Hernandez, U., and Law, J. (2015). Framing factors: The importance of context and the individual in understanding trust in human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Designing and Evaluating Social Robots for Public Settings*.

Edwards, S. and Lewis, C. (2012). ROS-Industrial: Applying the robot operating system (ROS) to industrial applications. In *IEEE Int. Conference on Robotics and Automation, ECHORD Workshop*.

Eimontaite, I., Gwilt, I., Cameron, D., Aitken, J.M., Rolph, J., Mokaram, S., and Law, J. (2016). Assessing graphical robot aids for interactive co-working. In *Proceedings of 7th International Conference on Applied Human Factors and Ergonomics*.

Fong, T., Kunz, C., Hiatt, L.M., and Bugajska, M. (2006). The human-robot interaction operating system. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 41–48. ACM.

Fong, T., Thorpe, C., and Baur, C. (2001). *Collaborative control: A robot-centric model for vehicle teleoperation*. Carnegie Mellon University, The Robotics Institute.

Fong, T., Thorpe, C., and Baur, C. (2003). Collaboration, dialogue, human-robot interaction. In *Robotics Research*, 255–266. Springer.

Kaupp, T., Makarenko, A., and Durrant-Whyte, H. (2010). Human–robot communication for collaborative decision makinga probabilistic approach. *Robotics and Autonomous Systems*, 58(5), 444–456.

Khansari-Zadeh, S.M. and Khatib, O. (2015). Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Autonomous Robots*, 1–25.

Kirchner, E.A., de Gea Fernandez, J., Kampmann, P., Schröer, M., Metzen, J.H., and Kirchner, F. (2015). Intuitive interaction with robots–technical approaches and challenges. In *Formal Modeling and Verification of Cyber-Physical Systems*, 224–248. Springer.

McAree, O., Aitken, J.M., and Veres, S.M. (2016). A model based design framework for safety verification of a semi-autonomous inspection drone. In *Control (CONTROL), UKACC International Conference on*.

Ososky, S., Schuster, D., Phillips, E., and Jentsch, F.G. (2013). Building appropriate trust in human-robot teams. In *2013 AAAI Spring Symposium Series*.

Pawar, V.M., Law, J., and Maple, C. (2016). Manufacturing robotics - the next robotic industrial revolution. Technical report, UK Robotics and Autonomous Systems Network. URL `http://hamlyn.doc.ic.ac.uk/uk-ras/sites/default/files/UK_RAS_wp_manufacturing_web.pdf`.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 5.

Shepherd, S. and Buchstab, A. (2014). Kuka robots on-site. In *Robotic Fabrication in Architecture, Art and Design 2014*, 373–380. Springer.

Sheridan, T.B. (1997). Eight ultimate challenges of human-robot communication. In *Robot and Human Communication, 1997. RO-MAN'97. Proceedings., 6th IEEE International Workshop on*, 9–14. IEEE.

Thomas, C., Busch, F., Kuhlenkoetter, B., and Deuse, J. (2011). Process and human safety in human-robot-interaction-a hybrid assistance system for welding applications. In *Intelligent Robotics and Applications*, 112–121. Springer.

Virga, S., Zettinig, O., Esposito, M., Pfister, K., Frisch, B., Neff, T., N.Navab, and Hennersperger, C. (2016). Automatic force-compliant robotic ultrasound screening of abdominal aortic aneurysms. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*.